Electronic Journal of Differential Equations, Vol. 2024 (2024), No. 53, pp. 1–17.
ISSN: 1072-6691. URL: https://ejde.math.txstate.edu, https://ejde.math.unt.edu
DOI: 10.58997/ejde.2024.53

DEEP LEARNING METHOD FOR FINDING EIGENPAIRS IN STURM-LIOUVILLE EIGENVALUE PROBLEMS

SEN ZHANG, JIAN ZU, JINGQI ZHANG

Abstract. Solving the eigenvalue problem for differential equations in inhomogeneous media poses a significant challenge across diverse scientific fields. While classical finite difference methods and finite element methods have produced numerous outcomes, they heavily rely on discretizing the computational domain, which can introduce complexities and limitations. In this study, we present an unsupervised neural network approach tailored for finding eigenpairs in Sturm-Liouville eigenvalue problems within inhomogeneous media. Our method introduces eigenvalues as trainable parameters, crafts a novel cost function, incorporates an adaptive hyper-parameter tuning strategy, and sequentially trains the eigenpairs. The simplicity, accuracy, and interpretability of our approach significantly expand its applicability across various domains. The method we present in this paper can easily tackle boundary value conditions with derivatives, resulting in orthogonal eigenfunctions. This is a very important advantage of deep learning methods that has not yet been noticed. Quantitative estimation of eigenpairs is given for the Sturm-Liouville eigenvalue problems. Furthermore, we extend the proposed methodology to tackle two-dimensional cases, periodic scenarios, demonstrating its versatility and broad potential.

1. Introduction

The Sturm-Liouville eigenvalue problem holds substantial significance across a diverse range of fields. It can be demonstrated that there are an infinite number of eigenvalues, each with a unique eigenfunction, and that these eigenfunctions form an orthonormal basis within a specific Hilbert space of functions. The adaptability of the Sturm-Liouville equation to various boundary value conditions allows for the precise representation of specific physical phenomena. For example, in the context of heat equations, Dirichlet conditions maintain a consistent temperature at the rod's extremities, while Neumann conditions simulate insulating environments with no heat transfer and negligible temperature gradients at the ends. Furthermore, more comprehensive boundary value conditions, such as the Robin condition, accommodate partially insulated boundaries.

In practical applications, it is also crucial to compute multiple orthogonal eigenvectors of the Sturm-Liouville eigenvalue problem. For instance, in the construction

2020 *Mathematics Subject Classification.* 34L16, 65F18, 65L15, 65N25, 68T07.
Key words and phrases. Eigenvalue problems; Sturm-Liouville problems; neural network; self-adjoint operator; inhomogeneous media.

Submitted December 2, 2023. Published September 12, 2024.

of a suitable Lyapunov function, there are instances where the need for several mutually orthogonal minimal eigenvectors arises, see [4]. Therefore, the quest for multiple minimal eigenpairs (i.e., eigenvalues and eigenvectors) for the Sturm-Liouville eigenvalue problem remains an enduring research pursuit.

Classical numerical methods employed in the quest for eigenpairs include finite difference methods [1, 17] and finite element methods [2]. The finite difference method, characterized by its simplicity and ease of comprehension, is a popular choice. However, it also exhibits inherent limitations. For instance, in the case of Sturm-Liouville eigenvalue problems with derivative boundary value conditions, such as Neumann boundary and Robin boundary, the finite difference approximation of the differential operator may not preserve self-adjointness. Consequently, the resulting eigenvectors are not orthogonal to each other, which contradicts the nature of Sturm-Liouville differential operators being self-adjoint and having orthogonal eigenvectors. The finite element method offers high computational accuracy and wide applicability. However, it requires manual specification of appropriate finite element units prior to computation, and its effectiveness is often reliant on individual expertise. It is a more complex method and may not be user-friendly for non-experts. Furthermore, it is worth noting that both finite difference methods and finite element methods yield numerical solutions, which lack analytical expressions and cannot be readily used for operations such as differentiation. Their applicability is relatively limited in situations requiring analytical expressions. The search for effective analytical approximations remains a significant scientific challenge.

In recent years, deep learning methods have rapidly advanced and integrated into diverse aspects of people's lives. Within the study of differential equations, their exceptional strengths and capabilities stand out. Chen et al. [6] proposed Neural ODE, which approximate the right-hand side function of differential equations using deep neural networks, achieving remarkable results. Currently, the most widely used approach is to approximate the solutions of differential equations using deep neural networks, with advantages like increased applicability, ease of understanding, and the ability to perform differentiable operations. Raissi et al. [19] introduced a deep learning framework known as 'Physics-informed neural networks' (PINN). Within this innovative framework, the physical equations themselves are seamlessly integrated into the network as constraints, thereby enabling the solution of partial differential equations (PDEs) without the necessity of a dataset. Pang et al. [18] proposed a fractional PINN (fPINN), addressing the challenge of automatic differentiation being unsuitable for fractional operators. Yang et al. [20] further advanced the field by developing a novel class of physics-informed generative adversarial networks, which effectively tackle forward, inverse, and mixed stochastic problems in a unified manner, relying on a limited number of scattered measurements. Lu et al. [16] have presented a Python library, named DeepXDE, specifically designed for PINNs. This library serves dual purposes: As an educational tool for classroom use and as a research tool for addressing computational science and engineering challenges.

For specific mathematical problems, such as identifying Hamiltonian systems, Mattheakis et al. [7] developed a Hamiltonian neural network (HNN) with the primary aim of conserving the symplectic structure of the Hamiltonian system. This allows for more accurate learning and prediction of the dynamic behavior of the system. Jin et al. [13] have developed symplectic networks (SympNets) to effectively learn the symplectic flow within Hamiltonian systems. Moreover, they [12] have also proposed Poisson neural networks (PNNs) which are specifically designed to learn Poisson systems and accurately capture the trajectories of autonomous systems, all derived directly from data. Lu et al. [15] proposed DeepONet, which has demonstrated its efficacy as a powerful tool for handling nonlinear operators using supervised data-driven methods. What's even more thrilling is the potential that emerges from merging DeepONet with the physics encoded by PINNs. This union opens up the possibility of achieving precise, real-time predictions in many fields with extrapolation capabilities. Zang et al. [21] presented an adversarial neural network rooted in partial differential equation weak solutions, offering a pathway for tackling high-dimensional partial differential equations. For further advancements related to physics-informed machine learning, we highly recommend delving into the comprehensive review article [14] for a deeper understanding.

The application of deep learning techniques in addressing eigenvalue problems has witnessed significant advancements. PINN, owing to its ease of implementation and ability to maintain the self-adjointness of the operator throughout the computational process, has been employed by scholars to identify orthogonal eigenpairs. Jin et al. [9] used PINN to solve the quantum problems related to finite wells, multiple finite wells, and hydrogen atom eigenvalues, and introduced a strong boundary value condition. However, their investigation was limited to one-dimensional homogeneous media under the Dirichlet boundary value condition. Holliday et al. [8] extended Jin's algorithm to quantum billiard eigenvalue problems, successfully identifying the eigenvalues and eigenfunctions of the stationary Schrödinger differential equation with Dirichlet boundary value conditions in two-dimensional domains. Ben-Shuaul et al. [5] have effectively employed PINN techniques to tackle the problem of finding the smallest eigenpairs for one-dimensional Sturm-Liouville boundary value problems. The strength of their approach lies in its ability to simultaneously identify multiple eigenpairs. However, their method, which incorporates the Rayleigh quotient for eigenvalue calculation, can be challenging for non-experts and may yield lower precision results. Additionally, their focus is specifically on scenarios involving the Laplace operator as the differential operator, disregarding inhomogeneous cases. Furthermore, their framework is limited to Dirichlet boundary value condition and does not account for derivative boundary situations. Unlike the Dirichlet boundary, the discrete matrix corresponding to the Sturm-Liouville differential operator under derivative boundaries is not self-adjoint when using classical numerical methods. This aspect underscores the advantages of leveraging neural networks for addressing such problems.

In this article, we consider how to find the smallest eigenpairs using deep learning methods. We introduce eigenvalues as trainable parameters, construct a novel cost function, add an adaptive hyper-parameter tuning strategy, and enhance the speed and accuracy of training by sequentially training eigenpairs. This setup allows us to compute a more extensive set of smallest orthogonal eigenpairs. By utilizing a weighted inner product space and adaptive hyper-parameters, we tackle the Sturm-Liouville eigenvalue problem under varying boundary value conditions in inhomogeneous media. The simplicity, accuracy, and interpretability of our method significantly broaden the applicability of the proposed approach. Quantitative estimation of eigenpairs is given for the Sturm-Liouville eigenvalue problems. Furthermore, we extend the proposed method to two-dimensional cases, periodic scenarios.

The organization of this paper is as follows. In Section 2, we introduce the problem model, construct a novel cost function, and present our algorithm. In Section 3, we give a quantitative estimation of eigenpairs. In Section 4, we compare our approach with previous algorithms, including state-of-the-art deep learning methods and classical numerical techniques. Section 5 presents extensive numerical experiments that demonstrate the efficiency and accuracy of our method in addressing the eigenvalue problem under various boundary conditions. Finally, in Section 6, we conclude the paper by summarizing our key findings and contributions.

2. Model problem

In this article, we aim to find the M smallest eigenvalues and eigenfunctions of the general Sturm-Liouville eigenvalue problems

$$(p(x)\varphi'(x))' + \lambda\sigma(x)\varphi(x) = 0, \tag{2.1}$$

with homogeneous boundary value conditions

$$a_1\varphi'(0) - b_1\varphi(0) = 0, \quad a_2\varphi'(\pi) + b_2\varphi(\pi) = 0, \quad a_i^2 + b_i^2 \neq 0,\ i = 1,2, \tag{2.2}$$

where $p \in C^1((0,\pi),\mathbb{R}^+)$ and $\sigma \in C((0,\pi),\mathbb{R}^+)$, λ is the eigenvalue and $\varphi \in C^2((0,\pi),\mathbb{R})$ is the corresponding eigenfunction. In some instances, p and σ are elastic coefficent and rock density coefficient respectively, see [3]. By a change of variable $x \to z$ given by $z = \int_0^x \sqrt{\frac{\sigma(s)}{p(s)}}ds$, equation (2.1) leads to

$$(\rho(z)\varphi'(z))' + \lambda\rho(z)\varphi(z) = 0, \tag{2.3}$$

where $\rho(z) = \sqrt{\sigma(z)p(z)}$ is the impedance function.

The mathematical theoretical results for the Sturm-Liouville eigenvalue problems (2.1) or (2.3) are relatively abundant. Let $\Omega = (0,\pi)$. Define the Sturm-Liouville operator in inhomogeneous media by

$$\mathcal{L}[\varphi] := -\frac{(p(x)\varphi'(x))'}{\sigma(x)} = \lambda\varphi, \quad \text{in } \Omega. \tag{2.4}$$

Define the weighted inner product by

$$(\varphi,\psi)_\sigma := \int_\Omega \varphi\psi^*\sigma \mathrm{d}x, \quad \varphi,\psi \in L^2(\Omega,\sigma\mathrm{d}x),$$

which induces the L^2 norm $\|\cdot\|_\sigma$. Note that when $\sigma \equiv 1$ on Ω, $\|\cdot\|_\sigma$ reduces to be the standard L^2 norm, which is denoted by $\|\cdot\|$. It is well known that $\mathcal{L}$ is a self-adjoint operator, the eigenvectors of $\mathcal{L}$ construct an orthonormal basis $(\varphi_k)_{k\in\mathbb{N}^+}$ of $L^2((0,\pi),\sigma\mathrm{d}x)$:

$$\mathcal{L}[\varphi_k] = \lambda_k\varphi_k, \quad k = 1,2,\dots, \tag{2.5}$$

where $(\lambda_k)_{k\in\mathbb{N}}$ is increasingly approach to $+\infty$ as $k \to +\infty$.

Finding analytical expressions for the eigenpairs of Sturm-Liouville problems in inhomogeneous media is highly challenging, so the deep learning solution we designed can only be compared with the classical numerical method. Using deep learning methods to solve eigenvalue problems offers numerous advantages. One such advantage is that the resulting neural network solutions are semi-analytical, meaning they are differentiable. The method is also simple, practical, and easily extensible, making it highly accessible to non-experts. Personally, one of the greatest benefits I have found is that for derivative boundary value cases, the neural network solutions obtained through this dimension-free approach are self-adjoint, aligning with the original problem. In contrast, classical numerical methods, such as finite difference method, often yield approximate solutions that do not satisfy self-adjointness, leading to numerical eigenfunctions that are not orthogonal, see [1].

Using neural networks to find the eigenpairs of $\{\lambda_k, \varphi_k\}$ in inhomogeneous media only requires starting with the problem itself. We gradually learn eigenpairs, starting from smaller values of k and progressing to larger ones. Inspired by the PINN, we integrate the equation equalities, boundary value conditions, normalization of eigenfunctions, and orthogonality among distinct eigenfunctions into the loss function. This approach ensures the feasibility and practical implementation of our method. Specifically, we introduce a parameterized form of the k-th eigenfunction, denoted by φ_{θ_k}. In this context, θ_k functions as the primary training parameter. Moreover, within our comprehensive framework, we incorporate the eigenvalue λ_k as an additional parameter that undergoes training alongside θ_k. For each k, when φ_{θ_k} learns the expected target through training, λ_k will also be automatically updated with the network to the eigenvalue corresponding to the eigenfunction network φ_{θ_k}. This design greatly reduces the complexity of the network and the difficulty of calculation, so that we only need to pay attention to the training of the eigenfunction network, and do not need to design an additional network to train the eigenvalue.

To optimize the neural network φ_{θ_k} with respect to its parameter θ_k, we define the loss function

$$J(\theta_k, \lambda_k) = \alpha J_{\rm eq}(\theta_k, \lambda_k) + \beta J_{\rm bdry}(\theta_k) + \gamma J_{\rm nor}(\theta_k) + \delta_k J_{\rm orth}(\theta_k), \tag{2.6}$$

where α, β, γ, δ_k are hyper-parameters that control the relative importance of each term. The individual terms are as follows:

$$J_{\rm eq}(\theta_k, \lambda_k) = \|\mathcal{L}[\varphi_{\theta_k}] - \lambda_k \varphi_{\theta_k}\|_\sigma^2 \tag{2.7}$$

enforces the constraints imposed by the equation.

$$J_{\rm bdry}(\theta_k) = \big|\mathcal{B}[\varphi_{\theta_k}]\big|^2 \tag{2.8}$$

accounts for the constraints imposed by the boundary value conditions.

$$J_{\rm nor}(\theta_k) = \frac{\big(\|\varphi_{\theta_k}\|_\sigma^2 - 1\big)^2}{\|\varphi_{\theta_k}\|_\sigma^2}, \tag{2.9}$$

is responsible for the normalization of the eigenfunction in the weight space. The denominator is included to prevent the norm of the eigenfunction from vanishing during the computation of higher eigenvalues.

$$J_{\rm orth}(\theta_k) = \sum_{j=1}^{k-1} \big|(\varphi_{\theta_k}, \varphi_{\theta_j})_\sigma\big|^2, \tag{2.10}$$

ensures orthogonality of the current eigenfunction with respect to previously computed eigenfunctions in the weight space.

Under this framework, the choice of δ_k is crucial for accurately computing higher-order eigenvalues λ_k. When the epoch is less than half of the total number of epochs, we set δ_k equal to $10 \cdot \lambda_{k-1}$ to ensure that the eigenvalues can correctly approach the desired values. However, when the epoch exceeds half of the total, we set δ_k to 1 to reduce the weight of the orthogonality term, thereby facilitating better computation of the loss associated with the equation term. Note that both the divisor of the third term and the addition of the δ_k are specially designed for obtaining the higher eigenpairs. For details of the process, see Algorithm 1.

Algorithm 1 Deep learning method for finding M eigenpairs

```
1: for $k = 1$ to $M$ do
2:     Set total number of epochs $N_{\text{epochs}}$
3:     Define hyper-parameters $\alpha$, $\beta$, $\gamma$, $\lambda_0$
4:     Initialize neural network $\varphi_{\theta_k}$ with parameters $\theta_k$ and $\lambda_k = \lambda_{k-1}$
5:     for epoch $= 1$ to $N_{\text{epochs}}$ do
6:         if epoch $< \frac{N_{\text{epochs}}}{2}$ then
7:             Set $\delta_k = 10 \cdot \lambda_{k-1}$
8:         else
9:             Set $\delta_k = 1$
10:        end if
11:        Compute loss function $J(\theta_k, \lambda_k)$ using Eq. (2.6)
12:        Compute gradients of $J(\theta_k, \lambda_k)$ with respect to $\theta_k$ and $\lambda_k$
13:        Update $\theta_k$ and $\lambda_k$ using gradient descent optimizer
14:        if convergence criterion is met then
15:            Break the loop
16:        end if
17:    end for
18:    Output optimized parameters $\theta_k$ and eigenvalue $\lambda_k$
19: end for
```

In the individual terms of cost function, $J_{\text{eq}}(\theta_k, \lambda_k)$ is calculated by

$$\|\mathcal{L}[\varphi_{\theta_k}] - \lambda_k \varphi_{\theta_k}\|_\sigma^2 = \frac{1}{N_k} \sum_{j=1}^{N_k} \sigma\left(x_r^{(j)}\right) \left(\mathcal{L}[\varphi_{\theta_k}](x_r^{(j)}) - \lambda_k \varphi_{\theta_k}(x_r^{(j)})\right)^2, \tag{2.11}$$

where $\{x_r^{(j)}\}$ are sets of inner points sampled uniformly in the entire domain Ω. The weight norm $\|\cdot\|_\sigma$ and weight inner product in J_{nor} and J_{orth} are calculated by the mid-point integral method. Indeed, our framework is applicable not only to 1-dimensional cases but also to 2-dimensional cases, as well as periodic scenarios. For the 1D case, the weight inner product is calculated by the mid-point integral method:

$$(\varphi_{\theta_k}, \varphi_{\theta_j})_\sigma = h \sum_{m=0}^{N-1} \sigma(x_h^{(m)}) \varphi_{\theta_k}(x_h^{(m)}) \varphi_{\theta_j}(x_h^{(m)}) \tag{2.12}$$

where $\{x_h^{(m)} | x_h^{(m)} = h/2 + m \cdot h\}$ are sets of integral points with $h = \pi/N$. The boundary value conditions for the eigenfunction φ are defined by

$$\mathcal{B}[\varphi] = [a_1\varphi'(0) - b_1\varphi(0),\, a_2\varphi'(\pi) + b_2\varphi(\pi)]. \tag{2.13}$$

In the experiments, the eigenfunction network φ_{θ_k} is set as a fully connected neural network with 4 hidden layers, each consisting of 32 neurons. Silu is used as the activation function. During training, we utilize the Adam optimizer to update the parameters of the neural network, with the initial learning rate of 0.01, which drops by 10% every 1000 epoch. A total of 50,000 epochs are set, ensuring sufficient time for network convergence. This computation is performed using the backpropagation algorithm implemented in PyTorch, a deep learning framework that facilitates automatic gradient computation.

3. Quantitative estimation of eigenpairs

We consider the Sturm-Liouville eigenvalue problem:

$$(\rho(x)\varphi'(x))' + \lambda\rho(x)\varphi(x) = 0, \tag{3.1}$$

with homogeneous boundary value conditions

$$a_1\varphi'(0) - b_1\varphi(0) = 0,\ a_2\varphi'(\pi) + b_2\varphi(\pi) = 0, \quad a_i^2 + b_i^2 \neq 0,\ i = 1, 2. \tag{3.2}$$

The boundary value conditions are divided into 6 cases:

(1) Dirichlet boundary value condition:
$a_1 = 0$, $b_1 > 0$,$a_2 = 0$, $b_2 > 0$;
(2) Neumann boundary value condition:
$a_1 > 0$, $b_1 = 0$, $a_2 > 0$, $b_2 = 0$;
(3) Dirichlet-Neumann boundary value condition:
$a_1 = 0$, $b_1 > 0$, $a_2 > 0$, $b_2 = 0$ or $a_1 > 0$, $b_1 = 0$, $a_2 = 0$, $b_2 > 0$;
(4) Mixed boundary value conditions I:
$a_1 = 0$, $b_1 > 0$, $a_2 > 0$, $b_2 \neq 0$ or $a_1 > 0$, $b_1 \neq 0$, $a_2 = 0$, $b_2 > 0$;
(5) Mixed boundary value conditions II:
$a_1 > 0$, $b_1 = 0$, $a_2 > 0$, $b_2 \neq 0$ or $a_1 > 0$, $b_1 \neq 0$, $a_2 > 0$, $b_2 = 0$;
(6) General boundary value condition:
$a_1 > 0$, $b_1 \neq 0$, $a_2 > 0$, $b_2 \neq 0$.

Assume that $\rho(x) \in H^2((0, \pi), \mathbb{R}^+)$. Let $z(x) = \sqrt{\rho(x)}\varphi(x)$, then the above equation can be transformed to

$$z''(x) + (\lambda - \eta_\rho(x))z(x) = 0,$$

where $\eta_\rho(x) = \frac{1}{2}\frac{\rho''}{\rho} - \frac{1}{4}(\frac{\rho'}{\rho})^2$ is assumed to be a real function in $L^2(0, \pi)$. The boundary value condition is transformed to a new one:

$$a_1 z'(0) - \Big(\frac{a_1}{2}\frac{\rho'(0)}{\rho(0)} + b_1\Big)z(0) = 0,\ a_2 z'(\pi) + \Big(b_2 - \frac{a_2}{2}\frac{\rho'(\pi)}{\rho(\pi)}\Big)z(\pi) = 0.$$

Let (Λ_k, φ_k) be the exact eigenpairs of (3.1)-(3.2). Let $(\lambda_k, \varphi_{\theta_k})$ be approximate eigenpairs calculated by our Neural Networks. Denote

$$\rho_0 := \operatorname{ess\,inf} \eta_\rho(x) \text{ and } \rho_1 := \rho_0 + \frac{2}{\pi}\int_0^\pi \{\eta_\rho(x) - \rho_0\}_+ \, dx,$$

where $\{h(x)\}_+ = \max\{h(x), 0\}$. Now, we give the estimation of Λ_k for Sturm-Liouville eigenvalue probems. The eigenvalue estimation for Dirichlet boundary value conditions has been well studied.

Lemma 3.1 ([22]). *The Sturm-Liouville problems* (3.1) *with Case (1) have the following inequalities:*

$$k^2 + \rho_0 \le \Lambda_k \le k^2 + \rho_1, \quad k \in \mathbb{N}^+.$$

The proof of above lemma can be seen in [22]. Notice that we have not assumed that $\rho_0 > 0$, so the case $\rho(x) \equiv \mathit{constant}$ is included.

Denote $\alpha_1 := \frac{\rho'(0)}{2\rho(0)} + \frac{b_1}{a_1}$ and $\alpha_2 := -\frac{\rho'(\pi)}{2\rho(\pi)} + \frac{b_2}{a_2}$. For other boundary value problems, we have the following result.

Lemma 3.2. *Let $k \in \mathbb{N}^+$. If $a_2 = 0$ and $\alpha_1 \geq 0$ (resp. $a_1 = 0$ and $\alpha_2 \geq 0$), the Sturm-Liouville problems* (3.1) *with Case (3) and (4) satisfy*

$$\Big(k-\frac{1}{2}\Big)^2 + \rho_0 \leq \Lambda_k \leq \Big(k-\frac{1}{2}\Big)^2 + \rho_1 + \frac{2\alpha_1}{\pi} \quad \Big(\mathit{resp.}\ \leq \Big(k-\frac{1}{2}\Big)^2 + \rho_1 + \frac{2\alpha_2}{\pi}\Big).$$

Proof. The case $a_2 = 0$ and $\alpha_1 = 0$ can be found in [11]. Now we prove that it still holds for the case $a_2 = 0$ and $\alpha_1 > 0$. It is obviously that $\Lambda_k > \rho_0$, $k \in \mathbb{N}^+$. We introduce the Prüfer transformation:

$$z_k = r\sin\theta, \quad z_k' = \sqrt{\Lambda_k - \rho_0}\, r\cos\theta,$$

where $r(x) > 0$. Assume that $z_k(x)$ has $(k-1)$ zeros in the interval $(0,\pi)$ denoted as $0 = \kappa_0 < \kappa_1 < \kappa_2 < \dots < \kappa_{k-1} < \kappa_k = \pi$. Define $\theta(\kappa_i) = i\pi, i = 1, 2, \dots, k-1$. Using the boundary conditions, we obtain

$$\theta(0) = \arctan\left(\frac{1}{\alpha_1}\sqrt{\Lambda_k - \rho_0}\right),\ \theta(\pi) = k\pi.$$

Firstly, we estimate the lower bound of Λ_k. It is not difficult to know that

$$\theta' = \sqrt{\Lambda_k - \rho_0} - \frac{(\eta_u(x) - \rho_0)\sin^2\theta}{\sqrt{\Lambda_k - \rho_0}} \leq \sqrt{\Lambda_k - \rho_0}.$$

Integrating both sides of the inequality over $(0,\pi)$, we obtain

$$\theta(\pi) - \theta(0) \leq \sqrt{\Lambda_k - \rho_0}\,\pi.$$

Since $\alpha_1 > 0$, utilizing the properties of $\arctan(x)$, we know that

$$\theta(0) = \arctan\Big(\frac{1}{\alpha_1}\sqrt{\Lambda_k - \rho_0}\Big) \in (0, \frac{\pi}{2}),$$

i.e.,

$$\theta(\pi) - \theta(0) > \Big(k - \frac{1}{2}\Big)\pi.$$

Thus, we conclude that

$$\Lambda_k \geq \Big(k - \frac{1}{2}\Big)^2 + \rho_0$$

Secondly, we prove the upper bound of Λ_k. Let $p_k = \sqrt{\Lambda_k - \rho_0}$. Then, as an increasing function of k, p_k approaches infinity and satisfies $p_k > 1/2$. Notice that

$$\lim_{k\to+\infty} p_k\Big(\arctan\big(\frac{p_k}{\alpha_1}\big) - \frac{\pi}{2}\Big) = -\alpha_1.$$

Define

$$f(p) := p\Big(\arctan\big(\frac{p}{\alpha_1}\big) - \frac{\pi}{2}\Big).$$

Taking the derivative of $f(p)$, we obtain

$$f'(p) = \arctan\big(\frac{p}{\alpha_1}\big) - \frac{\pi}{2} + \frac{\alpha_1 p}{p^2 + \alpha_1^2}.$$

and $\lim_{p\to+\infty} f'(p) = 0$. Since

$$f''(p) = \frac{2\alpha_1}{p^2 + \alpha_1^2} - \frac{2\alpha_1 p^2}{(p^2 + \alpha_1^2)^2} = \frac{2\alpha_1^3}{(p^2 + \alpha_1^2)^2} > 0,$$

we obtain $f'(p) < 0$, $p \in [\frac{1}{2}, +\infty)$. It implies that

$$\sqrt{\Lambda_k - \rho_0}\Big(\arctan\big(\frac{\sqrt{\Lambda_k - \rho_0}}{\alpha_1}\big) - \frac{\pi}{2}\Big) \geq -\alpha_1.$$

Hence

$$\theta(0) \geq \frac{\pi}{2} - \frac{\alpha_1}{\sqrt{\Lambda_k - \rho_0}}.$$

By the Prüfer transformation, we obtain

$$\begin{aligned}\theta' &= \sqrt{\Lambda_k - \rho_0}\cos^2\theta - \frac{(\eta_u(x) - \rho_0)\sin^2\theta}{\sqrt{\Lambda_k - \rho_0}} \\ &\geq \sqrt{\Lambda_k - \rho_0} - \frac{\{\eta_u(x) - \rho_0\}_+ \sin^2\theta}{\sqrt{\Lambda_k - \rho_0}} \\ &\geq \sqrt{\Lambda_k - \rho_0} - \frac{\{\eta_u(x) - \rho_0\}_+}{\sqrt{\Lambda_k - \rho_0}}.\end{aligned}$$

By integrating both sides over the interval $(0, \pi)$, we obtain:

$$\theta(\pi) - \theta(0) \geq \sqrt{\Lambda_k - \rho_0}\pi - \frac{\int_0^\pi \{\eta_u - \rho_0\}_+ \mathrm{d}x}{\sqrt{\Lambda_k - \rho_0}}.$$

Therefore,

$$k\pi - \frac{\pi}{2} + \frac{\alpha_1}{\sqrt{\Lambda_k - \rho_0}} \geq \sqrt{\Lambda_k - \rho_0}\pi - \frac{\int_0^\pi \{\eta_u - \rho_0\}_+ \mathrm{d}x}{\sqrt{\Lambda_k - \rho_0}}.$$

Defining $D := \Lambda_k - \rho_0$, $B := k - \frac{1}{2}$, $C := \frac{1}{\pi}(\int_0^\pi \{\eta_u(x) - \rho_0\}_+ \mathrm{d}x + \alpha_1)$, we obtain $D \leq B\sqrt{D} + C$, which implies

$$\sqrt{D} \leq (\sqrt{B^2 + 4C} + B)/2.$$

By inequalities

$$\sqrt{B^2 + 4C} = B\sqrt{1 + \frac{4C}{B^2}} \leq B(1 + \frac{2C}{B^2}) = B + \frac{2C}{B},$$

we obtain the upper bound

$$D \leq \frac{B^2 + 2B\sqrt{B^2 + 4C} + B^2 + 4C}{4} \leq B^2 + 2C.$$

Thus, we conclude that

$$\Lambda_k \leq \big(k - \frac{1}{2}\big)^2 + \rho_1 + \frac{2\alpha_1}{\pi}.$$

The aforementioned conclusion can be easily extend to the case where $a_1 = 0$ and $\alpha_2 \geq 0$. We omit the proof for brevity. $\square$

Lemma 3.3. *Assume that $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$. Then the Sturm-Liouville problems* (3.1) *with Case (2), (5) and (6) satisfy*

$$(k-1)^2 + \rho_0 \leq \Lambda_k \leq (k-1)^2 + \rho_1 + \frac{2\alpha_1}{\pi} + \frac{2\alpha_2}{\pi},\ k \in \mathbb{N}^+.$$

The above lemma can be proved by a similar argument to that of Lemma 3.2. Now, we give a rough estimation of λ_k obtained by the neural network method for the Sturm-Liouville eigenvalue problems.

Proposition 3.4. *Assume that $\rho_1-\rho_0\le 1$. If $\lambda_k\in[k^2+\rho_0,k^2+\rho_1]$, $k\in\mathbb{N}^+$, the Sturm-Liouville eigenvalue problems with Case (1) imply that*

$$|\lambda_k-\Lambda_k|\le 1\le \min_{j\ne k}\{|\lambda_k-\Lambda_j|\}.$$

Proof. By Lemma 3.1, we have $|\lambda_k-\Lambda_k|\le|\rho_1-\rho_0|\le 1$. For $j>k\ge 1$, we have

$$\begin{aligned}|\lambda_k-\Lambda_j|&\ge|\Lambda_j-\Lambda_k|-|\lambda_k-\Lambda_k|\\&\ge|j^2+\rho_0-k^2-\rho_1|-|k^2+\rho_1-k^2-\rho_0|\\&\ge|j^2-k^2|-2|\rho_1-\rho_0|\ge 1.\end{aligned}$$

The second inequality follows from Lemma 3.1, and the fourth inequality follows from the fact that $j^2-k^2\ge j+k\ge 3$. Similarly, for $k>j\ge 1$, we also conclude that

$$|\lambda-\Lambda_j|\ge 1.$$

Thus, for $j\ne k\in\mathbb{N}^+$, we have

$$|\lambda_k-\Lambda_k|\le|\rho_1-\rho_0|\le 1\le \min_{j}\{|\lambda_k-\Lambda_j|\}.\qquad\square$$

Proposition 3.5. *Assume that $a_2=0$, $\alpha_1\ge 0$ and $\rho_1-\rho_0+\frac{2\alpha_1}{\pi}\le\frac{2}{3}$ (resp. $a_1=0$, $\alpha_2\ge 0$ and $\rho_1-\rho_0+\frac{2\alpha_2}{\pi}\le\frac{2}{3}$). If $\lambda_k\in[(k-\frac12)^2+\rho_0,(k-\frac12)^2+\rho_1+\frac{2\alpha_1}{\pi}]$ (resp. $\lambda_k\in[(k-\frac12)^2+\rho_0,(k-\frac12)^2+\rho_1+\frac{2\alpha_2}{\pi}]$), $k\in\mathbb{N}^+$, the Sturm-Liouville eigenvalue problems with Case (3) and (4) imply that*

$$|\lambda_k-\Lambda_k|\le\frac{2}{3}\le \min_{j\ne k}\{|\lambda_k-\Lambda_j|\}.$$

By a similar argument to that of Proposition 3.4, we can prove Proposition 3.5 based on Lemma 3.2.

Proposition 3.6. *Assume that $\alpha_1\ge 0$, $\alpha_2\ge 0$ and $\rho_1-\rho_0+\frac{2\alpha_1}{\pi}+\frac{2\alpha_2}{\pi}\le\frac13$. If $\lambda_k\in[(k-1)^2+\rho_0,(k-1)^2+\rho_1+\frac{2\alpha_1}{\pi}+\frac{2\alpha_2}{\pi}]$, $k\in\mathbb{N}^+$, the Sturm-Liouville eigenvalue problems with Case (2), (5) and (6) imply that*

$$|\lambda_k-\Lambda_k|\le\frac{1}{3}\le \min_{j\ne k}\{|\lambda_k-\Lambda_j|\}.$$

By a similar argument to that of Proposition 3.4, we can prove Proposition 3.6 based on Lemma 3.3.

Denote $\mathcal{E}_k(x)=\mathcal{L}[\varphi_{\theta_k}](x)-\lambda_k\varphi_{\theta_k}(x)$, we can estimate the error of the eigenvalue by $\|\mathcal{E}_k(x)\|_\sigma$ and $\|\varphi_{\theta_k}\|_\sigma$.

Theorem 3.7 (Eigenvalue estimation). *If $|\lambda_k-\Lambda_k|\le\min_{j\ne k}\{|\lambda_k-\Lambda_j|\}$, $k\in\mathbb{N}^+$, we have*

$$|\lambda_k-\Lambda_k|\le\frac{\|\mathcal{E}_k(x)\|_\sigma}{\|\varphi_{\theta_k}\|_\sigma}.$$

Proof. Note that

$$\begin{aligned}\|\mathcal{L}[\varphi_{\theta_k}]-\lambda_k\varphi_{\theta_k}\|_\sigma^2&=(\mathcal{L}[\varphi_{\theta_k}]-\lambda_k\varphi_{\theta_k},\,\mathcal{L}[\varphi_{\theta_k}]-\lambda_k\varphi_{\theta_k})_\sigma\\&=\sum_{j=1}^\infty(\Lambda_j-\lambda_k)^2|(\varphi_{\theta_k},\,\varphi_j)_\sigma|^2\end{aligned}$$

$$\geq (\Lambda_k - \lambda_k)^2 \sum_{j=1}^{\infty} |(\varphi_{\theta_k}, \varphi_j)_\sigma|^2$$

Thus, we conclude that

$$|\lambda_k - \Lambda_k| \leq \frac{\|\mathcal{E}_k(x)\|_\sigma}{\|\varphi_{\theta_k}\|_\sigma}. \qquad \square$$

We define $P_k\varphi = (\varphi, \varphi_k)_\sigma \varphi_k$ and $P_k^\perp \varphi = \sum_{j\neq k}^{\infty} (\varphi, \varphi_j)_\sigma \varphi_j$. Obviously, $\varphi_{\theta_k} = P_k\varphi_{\theta_k} + P_k^\perp \varphi_{\theta_k}$. Next, we use $\|P_k^\perp \varphi_{\theta_k}|$ to evaluate the eigenfunction of our algorithm.

Theorem 3.8 (Eigenfunction estimation)**.** *Assume that $\rho_1 - \rho_0 < 1$. If $\lambda_k \in [k^2 + \rho_0, k^2 + \rho_1]$, $k \in \mathbb{N}^+$, the Sturm-Liouville eigenvalue problems with Case (1) satisfy*

$$\|P_1^\perp \varphi_{\theta_1}\|_\sigma \leq \frac{\|\mathcal{E}_1(x)\|_\sigma}{|\Lambda_2 - \lambda_1|},$$

or

$$\|P_k^\perp \varphi_{\theta_k}\|_\sigma \leq \frac{\|\mathcal{E}_k(x)\|_\sigma}{\min\{|\Lambda_{k+1} - \lambda_k|, |\Lambda_{k-1} - \lambda_k|\}}, \quad k \geq 2.$$

Proof. By Proposition 3.4, we have $|\lambda_k - \Lambda_j| \geq 1$, $k \neq j$. For $k = 1$, we obtain

$$\begin{aligned}
\|\mathcal{L}[\varphi_{\theta_1}] - \lambda_1 \varphi_{\theta_1}\|_\sigma^2 &= (\mathcal{L}[\varphi_{\theta_1}] - \lambda_1 \varphi_{\theta_1}, \mathcal{L}[\varphi_{\theta_1}] - \lambda_1 \varphi_{\theta_1})_\sigma \\
&= \sum_{j=1}^{\infty} (\Lambda_j - \lambda_1)^2 |(\varphi_{\theta_1}, \varphi_j)_\sigma|^2 \\
&\geq (\Lambda_2 - \lambda_1)^2 \sum_{j=2}^{\infty} |(\varphi_{\theta_1}, \varphi_j)_\sigma|^2 + (\Lambda_1 - \lambda_1)^2 |(\varphi_{\theta_1}, \varphi_1)_\sigma|^2 \\
&\geq (\Lambda_2 - \lambda_1)^2 \sum_{j=2}^{\infty} |(\varphi_{\theta_1}, \varphi_j)_\sigma|^2
\end{aligned}$$

Thus, we conclude that

$$\|P_1^\perp \varphi_{\theta_1}\|_\sigma \leq \frac{\|\mathcal{E}_1(x)\|_\sigma}{|\Lambda_2 - \lambda_1|}$$

For $k \geq 2$, we obtain

$$\begin{aligned}
\|\mathcal{L}[\varphi_{\theta_k}] - \lambda_k \varphi_{\theta_k}\|_\sigma^2 &= (\mathcal{L}[\varphi_{\theta_k}] - \lambda_k \varphi_{\theta_k}, \mathcal{L}[\varphi_{\theta_k}] - \lambda_k \varphi_{\theta_k})_\sigma \\
&= \sum_{j=1}^{\infty} (\Lambda_j - \lambda_k)^2 |(\varphi_{\theta_k}, \varphi_j)_\sigma|^2 \\
&\geq \min\{(\Lambda_{k+1} - \lambda_k)^2, (\Lambda_{k-1} - \lambda_k)^2\} \sum_{j=1, j\neq k}^{\infty} |(\varphi_{\theta_k}, \varphi_j)_\sigma|^2
\end{aligned}$$

Thus, we conclude that

$$\|P_k^\perp \varphi_{\theta_k}\|_\sigma \leq \frac{\|\mathcal{E}_k(x)\|_\sigma}{\min\{|\Lambda_{k+1} - \lambda_k|, |\Lambda_{k-1} - \lambda_k|\}} \qquad \square$$

Remark 3.9. By Proposition3.5 and 3.6, Theorem 3.8 can be easily extended to:

(1) Sturm-Liouville eigenvalue problems with Case (3) and (4), if $a_2 = 0$, $\alpha_1 \geq 0$, $\rho_1 - \rho_0 + \frac{2\alpha_1}{\pi} \leq \frac{2}{3}$ and $\lambda_k \in [(k - \frac{1}{2})^2 + \rho_0, (k - \frac{1}{2})^2 + \rho_1 + \frac{2\alpha_1}{\pi}]$ (resp. $a_1 = 0$, $\alpha_2 \geq 0$, $\rho_1 - \rho_0 + \frac{2\alpha_2}{\pi} \leq \frac{2}{3}$ and $\lambda_k \in [(k - \frac{1}{2})^2 + \rho_0, (k - \frac{1}{2})^2 + \rho_1 + \frac{2\alpha_2}{\pi}]$).
(2) Sturm-Liouville eigenvalue problems with Case (2), (5) and (6), if $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, $\rho_1 - \rho_0 + \frac{2\alpha_1}{\pi} + \frac{2\alpha_2}{\pi} \leq \frac{1}{3}$ and $\lambda_k \in [(k-1)^2 + \rho_0, (k-1)^2 + \rho_1 + \frac{2\alpha_1}{\pi} + \frac{2\alpha_2}{\pi}]$.

4. Comparison with previous algorithms

4.1. **Comparison with the state-of-the-art deep learning methods.** Consider the smallest eigenpairs problems

$$\varphi''(x) + \lambda\varphi(x) = 0, \tag{4.1}$$

with Dirichlet boundary value conditions

$$\varphi(0) = \varphi(\pi) = 0. \tag{4.2}$$

Ben-Shaul et al. [5] utilized a more intricate cost function to simultaneously identify the four smallest eigenpairs. In Table 4.1, Ben-Shaul^1 denotes the simultaneous training of six smallest eigenpairs using 60000 epochs with their approach. Ben-Shaul^2 represents the concurrent training of seven smallest eigenpairs using 70000 epochs. We find that their algorithm has a large error when $n = 6$, and they obtained incorrect results when $n = 7$. In contrast, our objective differs significantly. Our aim is to devise an easy-to-understand algorithm and adapt it for handling larger eigenvalues. Our method involves sequentially training 10 smallest eigenpairs, with each pair trained using 10000 epochs. Note that it is a fair comparison, the total number of epochs used in our method is equivalent to that of the other methods. We set the integral point to 500 and $l_r = 0.01$. The hyper-parameters are set as in [5]. We found that the runtime of our method is less than both Ben-Shaul^1 and Ben-Shaul^2, but it obviously has more accuracy, see Table 4.1. For our method, if we set 30000 epochs for each, 1000 inner points and 200 integral points, we will get more accurate eigenpairs.

Table 4.1. Comparison with Ben-Shaul et al.'s method.

Exact Value	Ben-Shaul^1	Ben-Shaul^2	Ours	(Runtime)
$\lambda_1 = 1$	0.9859	0.8639	1.0000	(39s)
$\lambda_2 = 4$	3.9042	1.0964	4.0000	(81s)
$\lambda_3 = 9$	8.8453	4.6499	8.9943	(149s)
$\lambda_4 = 16$	15.4377	8.5765	16.0017	(270s)
$\lambda_5 = 25$	24.4220	16.2587	24.9451	(338s)
$\lambda_6 = 36$	34.8055(1485s)	23.5614	35.9928	(404s)
$\lambda_7 = 49$	N/A	35.5628(2128s)	48.9332	(475s)
$\lambda_8 = 64$	N/A	N/A	63.9604	(552s)
$\lambda_9 = 81$	N/A	N/A	80.8455	(636s)
$\lambda_{10} = 100$	N/A	N/A	99.8336	(725s)

4.2. **Comparison with classical numerical methods.** The following finite difference method's discretization scheme is derived from reference [1]. When $a_1 = a_2 = 0$, we have Dirichlet boundary value condition.

$$\varphi(0) = \varphi(\pi) = 0. \tag{4.3}$$

If, at the internal grid points of the uniform grid

$$G = \{x_i; x_i = ih, i = 0, 1, \dots, n, n+1, h = \pi/(n+1)\}, \tag{4.4}$$

the operator $\mathcal{L}\varphi$ is approximated using central differences, with

$$\mathcal{L} = \begin{bmatrix} \frac{2}{h^2} + \eta_\rho(x_1) & -\frac{1}{h^2} & & & \\ -\frac{1}{h^2} & \frac{2}{h^2} + \eta_\rho(x_2) & -\frac{1}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h^2} & \frac{2}{h^2} + \eta_\rho(x_{n-1}) & -\frac{1}{h^2} \\ & & & -\frac{1}{h^2} & \frac{2}{h^2} + \eta_\rho(x_n) \end{bmatrix}. \tag{4.5}$$

Such a matrix is symmetric, ensuring the orthogonality of different eigenfunctions. The error of the k-th eigenvalue λ_k is about $O(k^4h^2)$. However, for other boundary value conditions, especially those involving derivatives($a_1 \neq 0$ or $a_2 \neq 0$), loss of symmetry brings additional difficulties when using classical numerical methods.

When $a_1 \neq 0$ or $a_2 \neq 0$, if we use finite difference method, we found that it is not symmetric, resulting in the eigenfunctions not being orthogonal, while the original differential operator is self-adjoint, ensuring the orthogonality of the real eigenfunctions. Specifically, the central difference eigenvalues λ_k, $k = 1, \dots, n+1$ of the above equation satisfy on the augmented grid

$$G^* = \{x_j; x_j = jh, j = -1, 0, 1, \dots, n, n+1, h = \pi/n\}, \tag{4.6}$$

the following eigenvalue problem of matrix

$$\mathcal{L} = \begin{bmatrix} \frac{2}{h^2} + \eta_\rho(x_0) & -\frac{\alpha}{h^2} & & & \\ -\frac{1}{h^2} & \frac{2}{h^2} + \eta_\rho(x_1) & -\frac{1}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h^2} & \frac{2}{h^2} + \eta_\rho(x_{n-1}) & -\frac{1}{h^2} \\ & & & -\frac{\beta}{h^2} & \frac{2}{h^2} + \eta_\rho(x_n) \end{bmatrix} \tag{4.7}$$

with

$$\alpha = \frac{2a_1}{a_1 + h\big(b_1 + \frac{a_1}{2}\frac{\rho'(0)}{\rho(0)}\big)} \quad \text{and} \quad \beta = \frac{2a_2}{a_2 + h\big(b_2 - \frac{a_2}{2}\frac{\rho'(\pi)}{\rho(\pi)}\big)}.$$

Obviously, it is not a diagonal matrix. Although the eigenvalues we calculated are correct, the corresponding eigenfunctions are not orthogonal, which limits the application of such a method.

Despite the fact that the study of the Dirichlet eigenvalue problem using deep learning methods is a hot topic, to our knowledge, there is no work on the boundary value condition involving derivatives using the deep learning method. Indeed, the method we present in this paper can easily tackle boundary value conditions with derivatives, resulting in orthogonal eigenfunctions. This is a very important advantage of deep learning methods that has not yet been noticed. Additionally, we don't require $\rho(x)$ to have a second-order derivative, which is an assumption required by the above method. Unlike classical numerical methods, our neural network solution provides an approximate analytical solution, in a certain sense, which allows us to obtain derivatives using automatic differentiation.

5. Numerical experiments

In this section, we will utilize the designed neural network algorithm to identify the smallest eigenpairs under various boundary value conditions. The numerical results obtained from the neural network algorithm will be compared to those from the classical numerical method, demonstrating the efficiency of our approach. Moreover, we illustrate that our proposed method can be seamlessly extended to handle two-dimensional cases and periodic scenarios.

Example 5.1. *The Sturm-Liouville problem with Neumann boundary value conditions:*

$$\begin{gathered} -(\rho(x)\varphi'(x))' = \lambda\rho(x)\varphi(x), \quad x \in (0,\pi), \\ \varphi'(0) = \varphi'(\pi) = 0, \end{gathered} \tag{5.1}$$

where $\rho(x) = 1 + 0.3\sin x$.

Table 5.1 shows the eigenvalues of the Sturm-Liouville Problem (5.1). $\widetilde{\lambda}_n$ represents the numerical results obtained via the finite difference method (4.5) with $h = \frac{\pi}{1000}$. $\widehat{\lambda}_n$ represents the numerical results obtained from the deep learning algorithm. We get the smallest eigenvalues and eigenfunctions simultaneously, see Figure 5.1.

Table 5.1. Eigenvalues of Sturm-Liouville Problem (5.1)

λ_n	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
$\widetilde{\lambda}_n$	$7.56 \cdot 10^{-6}$	1.1129	4.1098	9.1099	16.1119	25.1151	36.1193	49.1243	64.1302	81.1368
$\widehat{\lambda}_n$	$6.62 \cdot 10^{-6}$	1.1126	4.1082	9.1077	16.1057	25.1079	36.0993	49.1056	64.0550	81.0975

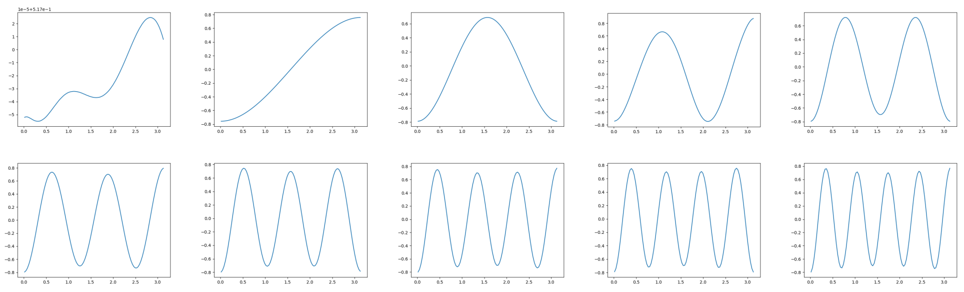

Figure 5.1. Eigenfunctions of the Sturm-Liouville Problem (5.1)

Example 5.1 shows that our neural network method is well adapted to the inhomogeneous media case, especially to the derivative boundary value case. The resulting eigenfunction is orthogonal and has broad prospects for use in other mathematical problems.

Example 5.2. Consider the periodic eigenpair problems:

$$\begin{gathered} -\varphi''(x) = \lambda\varphi(x), \quad x \in (0,\pi), \\ \varphi(0) = \varphi(\pi), \quad \varphi'(0) = \varphi'(\pi). \end{gathered} \tag{5.2}$$

Periodic phenomena are ubiquitous in physics. Regarding the periodic eigenvalue problem, where each eigenvalue corresponds to a pair of mutually orthogonal eigenvectors, to our knowledge, no scholar has yet studied how to use deep learning methods to find multiple smallest eigenpairs for such problems.

In Table 5.2, $\widehat{\lambda}_n$ represents the numerical results obtained from the deep learning algorithm, and λ_n represents the exact value of the eigenvalue. It shows that our algorithm can be easily applied to periodic boundary value cases with enough accuracy.

Table 5.2. Eigenvalues of periodic boundary value problems (5.2)

λ_n	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
λ_n	0	4	4	16	16	36	36	64	64	100
$\widehat{\lambda}_n$	0.0000	4.0000	3.9999	15.9998	16.0000	36.0002	36.0001	64.0007	63.9998	99.9998

Example 5.3. 2D Sturm-Liouville problem with Robin boundary value conditions [10]:

$$\begin{gathered}-\nabla(\rho(x,y)\nabla\varphi(x,y)) = \lambda\rho(x,y)\varphi(x,y), \quad (x,y)\in\Omega \\ \varphi(0,y)=\varphi_x(\pi,y)=\varphi(x,0)=\varphi_y(x,\pi)=0,\end{gathered} \tag{5.3}$$

where $\rho(x,y) = (1+0.3\sin x)(1+0.3\sin y)$.

In this example, the finite element method is used as the baseline to compare our Neural network method. The number of finite elements is set as 47905 by using the software of *Freefem++*. Usually, the finite elements method has very high accuracy. In Table 5.3, λ_n^* represents the numerical eigenvalues obtained by finite element method and $\widehat{\lambda}_n$ represents the numerical results obtained from our deep learning algorithm. In Figure 5.2, we show the mutual orthogonal eigenfunctions that are learned simultaneously with eigenvalues. Such eigenfunctions are hard to learn for a non-expert using classical numerical methods. But our method is beginner-friendly.

Table 5.3. Eigenvalues of 2D Sturm-Liouville problem (5.3)

λ_n	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
λ_n^*	0.4959	2.5067	2.5067	4.5175	6.5069	6.5069	8.5177	8.5177	12.5069	12.5069
$\widehat{\lambda}_n$	0.4959	2.5067	2.5067	4.5174	6.5068	6.5068	8.5176	8.5176	12.5075	12.5071

The final example demonstrates that our neural network possesses considerably good accuracy. Our neural network can be easily utilized, whereas the finite element method poses significant difficulties for non-experts. How to further improve accuracy of neural network methods remains an important question for future studies.

6. Conclusion

Deep learning techniques have shown significant advancements in addressing eigenvalue problems, particularly with the use of PINN (Physics-Informed Neural Networks) due to its ease of implementation and ability to maintain operator self-adjointness. PINN has been applied to solve quantum problems and Sturm-Liouville

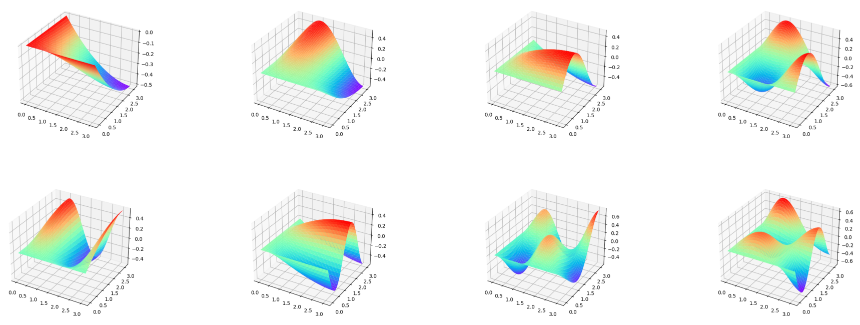

FIGURE 5.2. Eigenfunctions of 2D Sturm-Liouville problem with (5.3)

boundary value problems, but with limitations such as one-dimensional homogeneous media or specific boundary value conditions. This paper introduces a novel deep learning method to find the smallest eigenpairs by constructing a new cost function, incorporating adaptive hyper-parameter tuning, and sequentially training eigenpairs. The proposed approach extends to varying boundary value conditions in inhomogeneous media, two-dimensional cases, and periodic scenarios, demonstrating improved simplicity, accuracy, and interpretability. Quantitative estimation of eigenpairs is given for the Sturm-Liouville eigenvalue problems. Numerical experiments validate the method's efficiency and accuracy compared to previous algorithms [5].

REFERENCES

[1] R. Anderssen, F. De Hoog; On the correction of finite difference eigenvalue approximations for Sturm-Liouville problems with general boundary conditions, *BIT Numerical Mathematics*, **24** (1984), 401–412.

[2] A. Andrew, J. Paine; Correction of finite element estimates for Sturm-Liouville eigenvalues, *Numerische Mathematik*, **50** (1986), 205–215.

[3] V. Barbu, N. Pavel; Periodic solutions to nonlinear one dimensional wave equation with x-dependent coefficients, *Transactions of the American Mathematical Society*, **349** (1997), 2035–2048.

[4] K. Beauchard, M. Mirrahimi; Practical stabilization of a quantum particle in a one-dimensional infinite square potential well, *SIAM Journal on Control and Optimization*, **48** (2009), 1179–1205.

[5] I. Ben-Shaul, L. Bar, D. Fishelov, N. Sochen; Deep learning solution of the eigenvalue problem for differential operators, *Neural Computation*, **35** (2023), 1100–1134.

[6] R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud; Neural ordinary differential equations, *NeurIPS 2018*, **31** (2018).

[7] S. Greydanus, M. Dzamba, J. Yosinski; Hamiltonian neural networks, *NeurIPS 2019*, **32** (2019).

[8] E. Holliday, J. Lindner, W. Ditto; Solving quantum billiard eigenvalue problems with physics-informed machine learning, *AIP Advances*, **13** (2023), 085013.

[9] H. Jin, M. Mattheakis, P. Protopapas; Physics-informed neural networks for quantum eigenvalue problems, *In IJCNN at IEEE World Congress on Computational Intelligence* (2022).

[10] S. Ji; Periodic solutions of two-dimensional wave equations with x-dependent coefficients and Sturm-Liouville boundary conditions, *Nonlinearity*, **35** (2022), 5033.

[11] S. Ji, Y. Li; Periodic solutions to one-dimensional wave equation with x-dependent coefficients, *Journal of Differential Equations*, **229** (2006), 466–493.

[12] P. Jin, Z. Zhang, I. Kevrekidis, G. Karniadakis; Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks, *IEEE Transactions on Neural Networks and Learning Systems*, **34** (2022), 8271–8283.

[13] P. Jin, Z. Zhang, A. Zhu, Y. Tang, G. Karniadakis; SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems, *Neural Networks*, **132** (2020), 166–179.

[14] G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang; Physics-informed machine learning, *Nature Reviews Physics* **3** (2021), 422–440.

[15] L. Lu, P. Jin, G. Pang, Z. Zhang, G. Karniadakis; Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nature Machine Intelligence*, **3** (2021), 218–229.

[16] L. Lu, X. Meng, Z. Mao, G. Karniadakis; DeepXDE: A deep learning library for solving differential equations, *SIAM Review* **63** (2021), 208–228.

[17] J. Paine, R. Anderssen, F. De Hoog; On the correction of finite difference eigenvalue approximations for sturm-liouville problems, *Computing*, **26** (1981), 123–139.

[18] G. Pang, L. Lu, G. Karniadakis; fPINNs: Fractional physics-informed neural networks, *SIAM Journal on Scientific Computing*, **41** (2019), A2603–A2626.

[19] M. Raissi, P. Perdikaris, G. Karniadakis; Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational physics*, **378** (2019), 686–707.

[20] L. Yang, D. Zhang, G. Karniadakis; Physics-informed generative adversarial networks for stochastic differential equations, *SIAM Journal on Scientific Computing*, **42** (2020), A292–A317.

[21] Y. Zang, G. Bao, X. Ye, H. Zhou; Weak adversarial networks for high-dimensional partial differential equations, *Journal of Computational Physics*, **411** (2020), 109409.

[22] J. Zu; Approximate stabilization of one-dimensional Schrödinger equations in inhomogeneous media, *Journal of Optimization Theory and Applications*, **153** (2012), 758–768.

Sen Zhang
Center for Mathematics and Interdisciplinary Sciences, and School of Mathematics and Statistics, Northeast Normal University, Changchun, 130024, China
Email address: `zhangs832@nenu.edu.cn`

Jian Zu (corresponding author)
Center for Mathematics and Interdisciplinary Sciences, and School of Mathematics and Statistics, Northeast Normal University, Changchun, 130024, China
Email address: `zuj100@nenu.edu.cn`

Jingqi Zhang
Center for Mathematics and Interdisciplinary Sciences, and School of Mathematics and Statistics, Northeast Normal University, Changchun, 130024, China
Email address: `zhangjq906@nenu.edu.cn`